APPLICATION FOR UNITED STATES PATENT

FOR

METHOD AND APPARATUS FOR GATHERING QUEUE PERFORMANCE DATA

Inventors:	James S. Chapple
	Kalpesh D. Mehta
	Frank T. Hady

Prepared by:	Calvin E. Wells
	Reg. No. 43,256

METHOD AND APPARATUS FOR GATHERING QUEUE PERFORMANCE DATA

Field Of The Invention

[0001]   The present invention pertains to the field of computer systems. More particularly, this invention pertains to the field of gathering queue performance data in computer system components.

Background of the Invention

[0002]   In order to accomplish computer system performance tuning, it is important to be able to characterize various aspects of the computer system's behavior. One such aspect is queue utilization and behavior. A queue is a temporary storage space for instructions or data. Queues may be located throughout a computer system and are typically included in processors and chipset devices.

[0003]   Counters have been used in prior systems to collect data from physically inaccessible queues. The type of information that can be gathered by counters alone is limited to simple average metrics, such as throughput. Another measurement that can be made is the percentage of time that the queue is full. The amount of computer system performance tuning that can be performed with these simple metrics is limited. Therefore, an ability to collect more meaningful and useful data from queues is desirable.

1

Brief Description of the Drawings

[0004]   The invention will be understood more fully from the detailed description given below and from the accompanying drawings of embodiments of the invention which, however, should not be taken to limit the invention to the specific embodiments described, but are for explanation and understanding only.

[0005]   Figure 1 is a block diagram of a computer system including a queue and a queue performance data gathering unit.

[0006]   Figure 2 is a block diagram of a queue coupled to an event conditioning logic unit which is in turn coupled to a counter.

[0007]   Figure 3 is a block diagram of a queue performance data gathering unit that includes an event conditioning logic unit and two counters.

[0008]   Figure 4 is a timing diagram demonstrating one pass of a sample workload through the queue performance gathering unit of Figure 3 set up to take head-of-queue measurements with a threshold value of one.

[0009]   Figure 5 is a timing diagram demonstrating one pass of a sample workload through the queue performance gathering unit of Figure 3 set up to take head-of-queue measurements with a threshold value of two.

[0010]   Figure 6 is a timing diagram demonstrating one pass of a sample workload through the queue performance gathering unit of Figure 3 set up to take head-of-queue measurements with a threshold value of three.

[0011]   Figure 7 is a timing diagram demonstrating one pass of a sample workload through the queue performance gathering unit of Figure 3 set up to take head-of-queue measurements with a threshold value of four.

Detailed Description

[0012] The example embodiments described below take advantage of several signals generated at a queue, including a queue enter signal, a queue exit signal, and a queue not empty signal. These signals, in addition to allowing counters to both increment and decrement, provide a greater range of measurement possibilities than previously available. Embodiments are possible using more than one counter.

[0013] Figure 1 is a block diagram of a computer system including a queue 122 and a queue performance data gathering unit 124. The queue 122 and the queue performance data gathering unit 124 are included in a system logic device 120. The system logic device 120 receives commands from a processor 110. The commands for this example embodiment are temporarily stored in the queue 122. The commands are later delivered to an input/output hub 150. Many other types of communication may take place among the processor 110, the system logic device 120, and the input/output hub 150. The system logic device 120 is further coupled to a graphics controller 130 and a system memory 140.

[0014] The queue performance data gathering unit 124 receives signals from the queue 122 and performs data gathering operations. Some of the possible data gathering operations are discussed below.

[0015] Although this example embodiment discusses a single queue between the processor 110 and the input/output hub 150, other embodiments are possible using many more queues between many more interfaces. The queue 122 is meant to demonstrate one of many possible queues that may be included in a computer system.

[0016] Figure 2 is a block diagram of a queue 210 coupled to an event conditioning logic unit 220 which is in turn coupled to a counter 230. The queue 210 provides a queue enter signal 211, a queue exit signal 213, and a queue not empty signal 215 to the event conditioning logic unit 220. An inverted version of the queue not empty signal 215 (effectively a queue empty signal) is also presented to the event conditioning logic unit 220. The queue enter signal 211 is asserted for a clock period whenever a new entry has entered the queue 210. The queue exit signal 213 is asserted for a clock period whenever

3

an entry is discharged from the queue 210. The queue not empty signal is asserted whenever there is one or more entries in the queue 210.

[0017] The event conditioning logic unit 220 uses one or more of the signals from the queue 210 to generate an increment event signal 221 and a decrement event signal 223. The increment and decrement event signals 221 and 223 are coupled to a counter 230. The counter 230 increments in response to an assertion of the increment event signal 221 and decrements in response to an assertion of the decrement event signal 223.

[0018] The function of the event conditioning logic unit 220 may be programmable via software-visible registers. One possible function of the event conditioning logic 220 is to assert the increment event signal 221 whenever the queue enter signal is asserted and to assert the decrement event signal 223 whenever the queue exit signal 213 is asserted. If both the queue enter signal 211 and the queue exit signal 213 are both asserted, then the event conditioning logic does not assert the increment event signal 221 or the decrement event signal 223. By incrementing the counter 230 whenever a new entry enters the queue and by decrementing the counter 230 whenever an entry exits the queue, the counter will always contain a value that accurately represents the present fill level of the queue.

[0019] The event conditioning logic unit 220 may be programmed to perform other analysis tasks by using various combinations of the queue enter signal 211, the queue exit signal 213, the queue not empty signal 215, and the inverted version of the queue not empty signal 215 to trigger assertions of the increment event signal 221 and the decrement event signal 223.

[0020] Figure 3 is a block diagram of a queue performance data gathering unit that includes an event conditioning logic unit 320. This example embodiment is configured to perform head-of-queue measurements. Other configurations are possible to enable other types of queue analysis. The event conditioning logic unit 320 receives a queue enter signal 311, a queue exit signal 313, a queue not empty signal 315, and a queue empty signal 317 (an inverted version of the queue not empty signal 315) from a queue 310. The event conditioning logic unit 320 delivers an increment event A signal to a counter A and an increment event B signal to a counter B. The counter A delivers its value to one

4

input of a comparator 330. The comparator 330 receives its other input from a data register A which is programmable and visible to software. The comparator output signal 331 in this example becomes asserted when the counter value matches the contents of the data register A. The comparator output 331 is presented to the event conditioning logic unit 320. Other embodiments are possible where the comparator 330 asserts its output when the comparator 330 detects that the counter A value is less than the data register A value and also where the comparator 330 detects that the counter A value is greater than the data register A value. The functionality of the comparator 330 may be programmable.

[0021] The queue performance data gathering unit of Figure 3 further includes command register sets A and B which are each coupled to the event conditioning logic unit 320. The command registers A and B are also coupled to the data registers A and B, respectively.

[0022] For this example, a threshold level is written to the data register A. The event conditioning logic unit 320 is programmed to assert the increment event A signal whenever the queue not empty signal 315 is asserted. The event conditioning logic unit 320 is also programmed to assert the increment event B signal in response to an assertion of the comparator output signal 331. The counter A is reset whenever the queue exit signal 313 is asserted.

[0023] The head-of-queue measurement implemented in this example embodiment operates as follows. When there is an entry in the queue 310, the queue not empty signal 315 is asserted. The event conditioning logic unit 320 asserts the increment event A signal in response to the assertion of the queue not empty signal 315. The counter A is incremented in response to the assertion of the increment event A signal. The counter A continues to increment each clock period until the counter A is reset in response to an assertion of the queue exit signal 313.

[0024] During each clock period, the comparator 330 compares the counter A value with the threshold value stored in data register A. When the comparator 330 detects a match, the comparator output signal 331 is asserted resulting in an assertion of the increment event B signal. Counter B is incremented in response to the assertion of the increment event B signal. In this manner, the counter B tracks how many times during a

5

workload run a queue entry remained at the head of the queue for at least the number of clock periods indicated by the threshold value stored in data register A. At the end of the workload run, the counter B value can be accessed by reading the data register B which latches in the counter B value.

[0025] The operation of the queue data gathering unit of Figure 3 will be best understood by looking at several runs through a repeatable workload, as shown in the timing diagrams of Figures 4 through 7 below.

[0026] Figure 4 is a timing diagram demonstrating one pass of a sample workload through the queue performance gathering unit of Figure 3 with a threshold value of 1 stored in the data register A. During clock period 1, the queue not empty signal 315 and the queue exit signal 313 are not asserted, and the counters A and B are set to 0. Because the threshold level for this example is 1 and counter A contains 0, the comparator output signal 331 is not asserted.

[0027] During clock period 2, the queue not empty signal 315 becomes asserted and counter A is incremented. The counter A value now matches the threshold value and the comparator output signal 331 is accordingly asserted. The assertion of the comparator output signal 331 results in the incrementing of the counter B, which then contains 1.

[0028] During clock periods 3 and 4 the queue not empty signal 315 remains asserted and the queue exit signal 313 remains deasserted. Therefore, counter A continues to increment. The counter A values do not match the threshold value, so the counter B value remains at 1.

[0029] During clock period 5, the queue exit signal 313 becomes asserted while the queue not empty signal 315 remains asserted, indicating that there is a new entry at the head of the queue 320. The counter A is reset and incremented to the value 1. The counter A value then matches the threshold value stored in the data register A and the comparator output signal 331 becomes asserted by the comparator 330. The assertion of the comparator output signal 331 results in the incrementing of the counter B, which then contains the value 2.

[0030] During clock period 7, the queue exit signal 313 is again asserted, resulting in the counter A being reset to 0. The queue not empty signal 315 is also not asserted, so the

6

counter A is not incremented. The queue not empty signal 315 becomes asserted again during clock period 9 and remains asserted until clock period 27. Therefore, during clock periods 9 through 26 the counter A is incremented once each clock period. The queue exit signal 313 is asserted during clock periods 10, 14, 18, 20, 22-23, and 27 resulting in the counter A being reset during those clock periods. The comparator output signal 331 becomes asserted during clock periods 9-10, 14, 18, 20, and 22-23. At the end of the workload run the counter B indicates that queue entries remained at the head of the queue greater than or equal to one clock period (the threshold value for this example) 9 times during the workload run.

[0031]   By changing the threshold value and rerunning the same workload, a histogram of head-of-queue times can be established. The following figures follow the same workload as that discussed above in connection with Figure 4.

[0032]   Figure 5 is a timing diagram demonstrating one pass of the sample workload through the queue performance gathering unit of Figure 3 with a threshold value of 2. Because the workload for this example is the same as that used for the example of Figure 4, the queue not empty 315, queue exit 313, and Counter A waveforms of Figure 5 are identical to those of Figure 4. Because the threshold value is 2 for this example, the comparator output signal 331 becomes asserted when the Counter A value reaches 2. The Counter B value is incremented in response to the assertions of the comparator output signal 331. At the end of the workload run, the Counter B indicates that queue entries remained at the head of the queue greater than or equal to 2 clock periods (the threshold value for this example) 7 times during the workload run.

[0033]   An additional piece of data can now be determined after the second workload run. It is now known that queue entries remained at the head of the queue for exactly one clock period a total of 2 times during the workload run. This can be determined by observing the facts that the first run with a threshold of 1 resulted in a counter B value of 9 and that the second run with a threshold of 2 resulted in a counter B value of 7. The difference between the counter B value for the first run and the counter B value for the second run gives the exact number of times that queue entries remained at the head of the queue for exactly one clock period (the threshold value for the first run).

7

[0034]   Figure 6 is a timing diagram demonstrating one pass of the sample workload through the queue performance gathering unit of Figure 3 with a threshold value of 3. Because the workload for this example is the same as that used for the examples of Figures 4 and 5, the queue not empty 315, queue exit 313, and Counter A waveforms of Figure 6 are identical to those of Figures 4 and 5. Because the threshold value is 3 for this example, the comparator output signal 331 becomes asserted when the Counter A value reaches 3. The Counter B value is incremented in response to the assertions of the comparator output signal 331. At the end of the workload run, the Counter B indicates that queue entries remained at the head of the queue greater than or equal to 3 clock periods (the threshold value for this example) 4 times during the workload run.

[0035]   A further piece of data can now be determined after the third workload run. It is now known that queue entries remained at the head of the queue for exactly 2 clock periods a total of 3 times during the workload run. This can be determined by observing the facts that the second run with a threshold value of 2 resulted in a counter B value of 7 and that the third run with a threshold value of 3 resulted in a counter B value of 4. The difference between the counter B value for the second run and the counter B value for the third run gives the exact number of times that queue entries remained at the head of the queue for exactly two clock periods (the threshold value for the second run).

[0036]   Figure 7 is a timing diagram demonstrating one pass of the sample workload through the queue performance gathering unit of Figure 3 with a threshold value of 4. Because the workload for this example is the same as that used for the examples of Figures 4, 5, and 6, the queue not empty 315, queue exit 313, and Counter A waveforms of Figure 7 are identical to those of Figures 4, 5, and 6. Because the threshold value is 4 for this example, the comparator output signal 331 becomes asserted when the Counter A value reaches 4. The Counter B value is incremented in response to the assertions of the comparator output signal 331. At the end of the workload run, the Counter B indicates that queue entries remained at the head of the queue greater than or equal to 4 clock periods (the threshold value for this example) 3 times during the workload run.

[0037]   An additional piece of data can now be determined after the fourth workload run. It is now known that queue entries remained at the head of the queue for exactly

8

three clock periods a total of 1 time during the workload run. This can be determined by observing the facts that the third run with a threshold of 3 resulted in a counter B value of 4 and that the fourth run with a threshold of 4 resulted in a counter B value of 3. The difference between the counter B value for the third run and the counter B value for the fourth run gives the exact number of times that queue entries remained at the head of the queue for exactly three clock periods (the threshold value for the third run).

[0038]   specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than in a restrictive sense.

[0039]   Reference in the specification to "an embodiment," "one embodiment," "some embodiments," or "other embodiments" means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least some embodiments, but not necessarily all embodiments, of the invention. The various appearances of "an embodiment," "one embodiment," or "some embodiments" are not necessarily all referring to the same embodiments.